Vol.2017-IOT-36 No.12
2017/3/3

# センサネットワークのセンサ端末群を インターネット上の Wiki ページで制御する IoT システムの試作

山之上　卓[†1]　　羅　牧野[†2]

**概要**：インターネット上の Wiki ページにより、センサネットワークのセンサ端末群を制御する IoT システムの試作について述べる。センサ端末群から得られるデータを管理者が観察している最中に、特定のセンサ端末のみ、データを取得する時間間隔を途中で変えたくなる場合がある。また、センサの出力が、ある値を超えた時だけ、インターネット側のサーバにデータを出力している場合、その値を変えて調整したくなる場合がある。本システムは、このような要求を、物理的に、センサ端末がある場所に行かなくても、世界中、どこからでも、そのセンサ端末群を制御している Wiki ページの記述を変えるだけで、実現しようとするものである。

**キーワード**：Wiki, Bot, parallel programming, dynamic programming

# Experimental implementation of an IoT system which controls sensor terminals of a sensor network by a Wiki page on the Internet

TAKASHI YAMANOUE[†1] , LUO MUYE[†2]

***Abstract***: An experimental implementation of an IoT system is shown. Sensor terminals of the IoT system are controlled by a Wiki page on the Internet. When the manager of an IoT system is observing data from sensor terminals of the IoT system, the manager often wants to change the interval of data acquisition term of specific sensor terminals of the IoT system. When a sensor terminal is configured to report its sensor value when the value goes over or goes under a designated value, the manager often wants to change the designated value. This IoT system realizes such needs by enabling sensor terminals can be controlled by a Wiki page, instead of control sensor terminals by going to the place of the terminals.

***Keywords***: Wiki, bot, parallel programming, dynamic programming

## 1. はじめに

Wiki[17]はブラウザを使って, 簡単に相互にリンクされた Web ページを作ったり, その Web ページを編集したりすることができる Web サイトであり, 協同作業したり, 情報の共有をしたりすることを効果的に行うためによく利用されている[2].

もし, Wiki が人間にとって優しいものであるのなら, Wiki は機械にとっても優しいものであるはずである. もし機械が自動的に Wiki ページのデータを読んだり書いたりすることができるなら, 人間はより多くの有用な情報を得ることができるはずである. また, Wiki ページを通して, 人間が簡単に機械を制御することも可能になるはずである. 人間-機械間の通信だけでなく, 機械間の通信も可能になるはずである. これらのことにより, 機械が Wiki のページを読み書きできるようになることにより, Wiki は従来の人間だけが使うものより有益なものになるはずである. 例えば, よく利用されている Wiki がセンサネットワークのセンサを接続するために利用できるようになれば, 我々自身のセンサネットワークを簡単に構築できるようになるはずである(図 1).

これらのことを確認するため, 我々は Wiki を使った IoT システムを開発している. このシステムを実現するため, 今までに, Android[15]と Arduino[16]を組み合わせて構成された遠隔端末や, Raspberry Pi[18] で構成された遠隔端末を開発した. 一般的な Wiki ソフトウェアを IoT システムのデータベースとして利用することにより, 特定の Web サイトや Web ソフトウェアに縛られることなく, 広くセンサネットワークシステムを利用することが可能になる.

PukiWiki[17]は日本で一般的に使われている Wiki ソフトウェアの一つである. 我々は PukiWiki のページを読み書き可能にし, なおかつ, PukiWiki のページからも起動可能な Java の API を開発している[4][6]. この API を, PukiWiki-Java Connector (PJC) と呼んでいる. PJC を改造することにより, Raspberry Pi で PukiWiki のページの読み書きを可能にする API, *PukiWiki-Java Connector for Pi,* も作成している[14].

†1 福山大学
Fukuyama University
†2 鹿児島大学大学院理工学研究科
Graduate School of Science and Engineering Kagoshima University

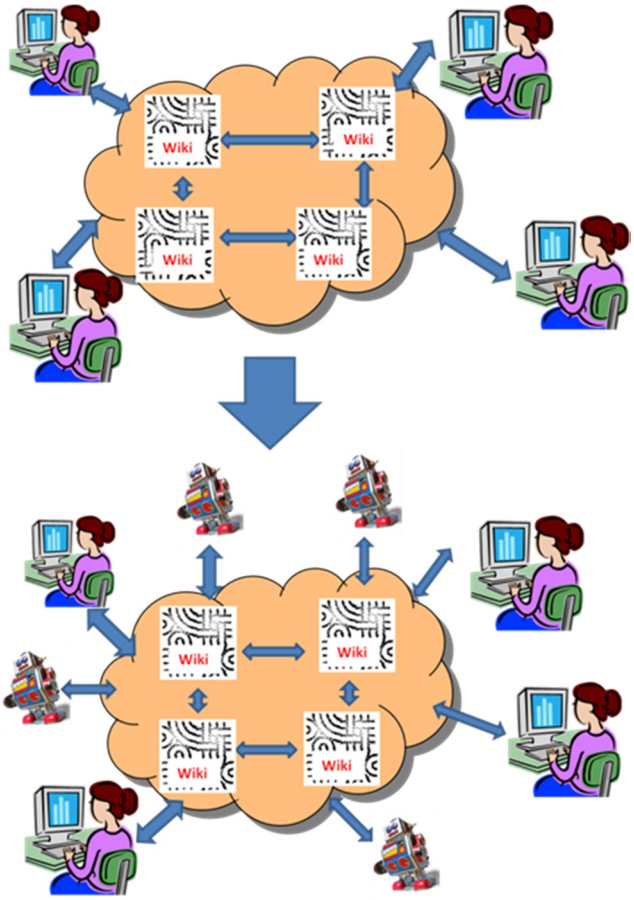


図 1. 人間も機械も利用できる Wiki ネットワーク

我々の IoT システムは PukiWiki の Wiki ページのネットワークと, Wiki ページを読み書きする端末群で構成されている. 端末は, Wiki のページを PJC や PJC for Pi を使って読み書きしたり, センサからデータを入力したり, アクチュエータにデータを送って動かしたりする. この端末群の端末は一種の Bot であり, Wiki に記載された設定やプログラムによって動作する. また, この Bot の一部を改変することにより, 情報セキュリティを脅かす悪性 Bot の所在を検知したり悪性 Bot の通信を横取りしたりして, 悪性 Bot を捕まえるシステムの開発も行っている[7][9]. これらの Bot のプログラミング環境は, ノイマン型コンピュータがプログラムもデータもメモリ上に記述されているのと同様に, プログラムもデータも Wiki ページ上に記述される. これを「Wiki ページ型」と呼ぶこととする. 我々は Wiki ページ型プログラミングの特徴を使い, 可用性の高い分散システムの構築の可能についても検討を行っている[12][13].

今回, IEEE802.15.4 準拠のトランシーバを備えた小型無線センサ端末.(TWE-Lite)[11]で構成された無線センサネットワーク(WSN)を利用することにより, 我々の IoT システムを拡張することについて述べる.

IoT システムの利用場面において, センサ端末群から得られるデータを管理者が観察している最中に、特定のセンサ端末のみ、データを取得する時間間隔を途中で変えたくなる場合がある。また、センサの出力が、ある値を超えた時だけ、インターネット側のサーバにデータを出力している場合、その値を変えて調整したくなる場合がある。本システムは、このような要求を、物理的に、センサ端末がある場所に行かなくても、世界中、どこからでも、そのセンサ端末群を制御している Wiki ページの記述を変えるだけで、実現しようとするものである。

## 2. IoT システム全体の概要

図 2 に、WSN を利用した IoT システム全体の概要図を示す。この IoT システムは、WSN、Bot/Gateway、Wiki ネットワークによって構成されている。

WSN は、この無線センサネットワークを含む IoT システムの中で、実世界とのインターフェースを担う。Bot/Gateway は、無線センサネットワークと Wiki ネットワークを接続する役割などを担う。Wiki ネットワークは、本 IoT システムのデータベースとしての役割を担う。

WSN のセンサ端末に接続されたセンサからデータが入力さると、WSN と Bot/Gateway を通じて Wiki ネットワークの Wiki ページに書き込まれる。また、Wiki ネットワークの Wiki ページに書かれたコマンドやデータが、Bot/Gateway と 無線 LAN を通じて、センサ端末に送られ、そのコマンドやデータによって、モータなどのアクチュエータが制御される。

Wiki には、Gateway 機能を持たない Bot が接続されることもある。このような Bot は、Wiki ページに書かれたデータを解析し、その結果を Wiki ページに書き戻す機能を持つ。このような Gateway 機能を持たない Bot を複数個組み合わせて、IoT のデータ解析を並列計算によって行うことも可能になる。我々はデータ解析で良く利用される R のプログラムを実行できる Bot も開発している。

Bot/Gateway や Bot は、Wiki ネットワークの Wiki ページに書かれたコマンドやプログラムを定期的に実行する。もし、それらのコマンドやプログラムの中に、無線センサネットワークのセンサ端末の設定を変更するコマンドが含まれていたら、そのコマンドが Bot/Gateway を通じて、対象のセンサ端末に送信され、そのセンサ端末の設定が変更される。

Bot/Gateway や無線センサネットワークは、インターネットに直接接続されている必要はなく、NAT の内側の LAN にこれらが設置されていても、その LAN からインターネット上の Web ページが閲覧できれば、本 IoT システムは動作可能である。このとき、Bot/Gateway は、Wiki ネットワークの Wik ページを読み書き可能であるが、インターネット側から Bot/Gateway に侵入することは困難になる。この性質は、本 IoT システムのセキュリティを強化するためにも有用である。

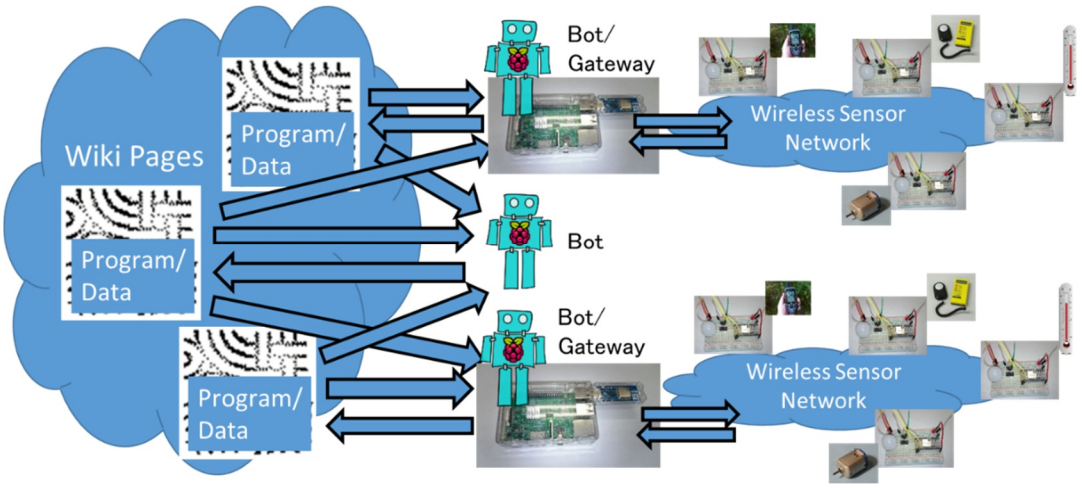


図 1．本無線センサネットワークを利用した IoT システム全体の概要

## 3. コマンドで端末の設定変更を可能とする WSN

WSN のセンサ端末として、モノワイヤレス株式会社の TWE-Lite 無線マイコンを利用した。図 2 に TWE-Lite を使って実現したセンサ端末の例を示す。ここで右側の DIP パッケージの部品が TWE-Lite 本体であり、この中に、IEEE802.15.4 準拠のトランシーバ、32bit マイコン、メモリ、周辺回路が組み込まれている。左側の白く丸い部品は PIR(Passive Infra-RED, 受動的赤外線センサ)であり、人や動物の動きを感知することができる。

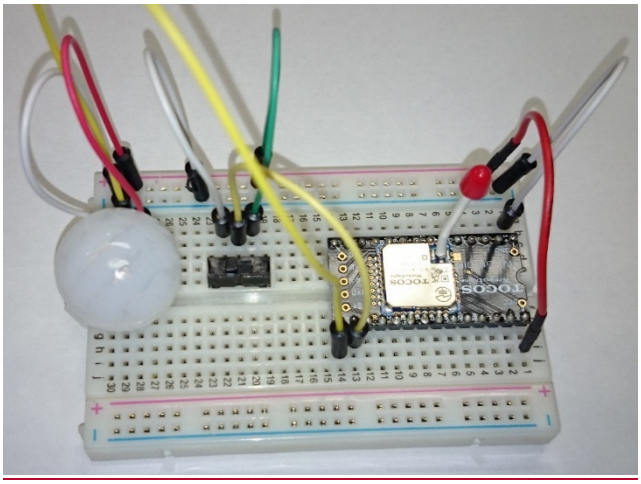

図 2. TWE-Lite を使って実現したセンサ端末の例

Twe-Lite のマイコンのプログラムは、フラッシュメモリ上に書き込まれており、上書きして書き換えることができる。

TWE-Lite 間の距離が長くて直接電波が届きにくい場合、間に中継用 TWE-Lite を置くことにより、到達距離を延ばすことができる。親ノードと沢山の子ノードと、その間の中継ノードを設置することにより、広範囲なデータを収集することも可能である。このとき、親ノードと子ノード間が通信するときの中継ノードの選定は、設定により、自動的に行わせることができる。

TWE-Lite のプログラムを開発するための、SDK と例題プログラムが公開されており、これを使うことにより、あらかじめ用意されたプログラム以外のプログラムを開発することが可能になっている。

本無線センサネットワークシステムの無線センサ端末のプログラム (以下、本プログラム) は、この SDK を使い、公開されていたシリアル通信専用アプリのプログラムを書き換えることにより、開発した。本プログラムを開発することにおり、事前に用意されていたプログラムでは不可能であった、TWE-Lite の、遠隔地からの、細かな設定変更や制御が可能になった。

本プログラムは、TWE-Lite が、無線やシリアル通信で送られてきた以下のコマンドを解釈実行できるようにするものである。なお、本無線センサネットワークにおいて、センサ端末間の通信は、すべてブロードキャストによって行われるものとする。

● get <id> port <analog/digital input port>.

このコマンドを受け取ったとき、もし、自分(TWE-Lite 端末)が id で指定された端末であった場合、<analog/digital-input-port>で指定されたポートの値を、すぐに、このコマンドを送信した端末に返送する。ここで

<id>=*|a32=<hex>|id=<number>|handle=<strconst>

である。*はこのコマンドを送信した端末以外のすべての端末を表す。<hex>は 8byte の端末識別子を 16 進で表したものである。<number>は TWE-Lite で規定されている 2byte の論理端末番号である。<strconst>は、この端末に名付けられた名前を表す文字列である。

例えば、

get id=1 port DI

は、もじ自分の論理端末番号が 1 であるなら、デジタル入力ポート DI1 から、デジタルデータ(1 か 0)を読み込んで、その値をすぐに、このコマンドを送信した端末に返すことを表す。

この端末から、コマンドを送信した端末に値を返送するのは、以下のコマンドをこの端末から送信することによって実現する。

return a32=<送信先端末識別子>, from=<自端末識別子>,
port=<analog/digital-input-port>, v=<value>, event=get

ここで、<送信先端末識別子>と<自端末識別子>は、8byte の端末識別子を 16 進で表したものである。

● set <id> sendif <analog/digital-input-port>
interval <time-msec>.

このコマンドを受け取ったとき、もし、自分(TWE-Lite 端末)が id で指定された端末であった場合、<analog/digital-input-port>で指定されたポートの値を、<time-msec>で指定された時間間隔で、このコマンドを送った端末に繰り返し送信することを表す。

この端末から、このコマンドによって設定された間隔で、このコマンドを送信した端末に値を返送するのは、以下のコマンドをこの端末から送信することによって実現する。

return a32=<送信先端末識別子>, from=<自端末識別子>,
port=<analog/digital-input-port>,
v=<value>, event=interval

ここで、<送信先端末識別子>と<自端末識別子>は、8byte の端末識別子を 16 進で表したものである。

● set <id> sendif <analog/digital-input-port>
changed <value>

このコマンドを受け取ったとき、もし、自分(TWE-Lite 端末)が id で指定された端末であった場合、<analog/digital-input-port>で指定されたポートの値が、以前に送信したときとくらべて、<value>の値を超えて変化したときに、その値を、このコマンドを送った端末に送信することを表す。<analog/digital-input-port>がデジタル入力端子で、内部状態 mode の値が 1 である場合、この入力端子の値が、1 から 0、または 0 から 1 に変化したら、変化後の値を、このコマンドを送った端末に送信する。Mode の値が 2 である場合、一定間隔（この後の送信間隔を設定するコマンドで設定された間隔）で、ポートの値が加算されていき、その加算された値が <value>を超えたときに、加算された値を、このコマンドを送った端末に送信する。

<analog/digital-input-port>がアナログ入力端子で、内部状態 mode の値が 1 である場合、この入力端子の値が、以前に送信したときと比べて、<value>の値を超えて変化したとき、その値を、このコマンドを送った端末に送信する。Mode の値が 2 である場合、上の interval 設定コマンドで設定された間隔で、ポートの値が加算されていき、その加算された値が <value>を超えたときに、加算された値を、このコマンドを送った端末に送信する。

この端末から、コマンドを送信した端末に値を返送するのは、以下のコマンドをこの端末から送信することによって実現する。

return a32=<送信先端末識別子>, from=<自端末識別子>,
port=<analog/digital-input-port>, v=<value>, event=changed

ここで、<送信先端末識別子>と<自端末識別子>は、8byte の端末識別子を 16 進で表したものである。

● set <id> sendif <analog/digital-input-port>
overTheLevel <value>

このコマンドを受け取ったとき、もし、自分(TWE-Lite 端末)が id で指定された端末であった場合、<analog/digital-input-port>で指定されたポートの値が、<value>の値を超えたときに、その値を、このコマンドを送った端末に送信することを表す。<analog/digital-input-port>がデジタル入力端子で、内部状態 mode の値が 1 である場合、このコマンドによるデータ送信は行われない。Mode の値が 2 である場合、一定間隔（この後の送信間隔を設定するコマンドで設定された間隔）で、ポートの値が加算されていき、その加算された値が <value>を超えたときに、加算された値を、このコマンドを送った端末に送信する。

<analog/digital-input-port>がアナログ入力端子で、内部状態 mode の値が 1 である場合、この入力端子の値が、<value>の値を超えたとき、その値を、このコマンドを送った端末に送信する。Mode の値が 2 である場合、一定間隔で、ポートの値が加算されていき、その加算された値が <value>を超えたときに、加算された値を、このコマンドを送った端末に送信する。

この端末から、コマンドを送信した端末に値を返送するのは、以下のコマンドをこの端末から送信することによって実現する。

return a32=<送信先端末識別子>, from=<自端末識別子>,
port=<analog/digital-input-port>, v=<value>,
event=overTheLevel

ここで、<送信先端末識別子>と<自端末識別子>は、8byte の端末識別子を 16 進で表したものである。

● set <id> sendif <alalog//digital-input-port>
underTheLevel <value>

このコマンドを受け取ったとき、もし、自分(TWE-Lite 端末)が id で指定された端末であった場合、<analog/digital-input-port>で指定されたポートの値が、<value>の値を下回ったとき、その値を、このコマンドを送った端末に送信することを表す。<analog/digital-input-port>がデジタル入力端子の場合はなにもしない。

<analog/digital-input-port>がアナログ入力端子で、内部状態

mode の値が 1 である場合、この入力端子の値が、<value>の値を下回ったとき、その値を、このコマンドを送った端末に送信する。Mode の値が 2 である場合、なにもしない。この端末から、コマンドを送信した端末に値を返送するのは、以下のコマンドをこの端末から送信することによって実現する。

return a32=<送信先端末識別子>, from=<自端末識別子>,
port=<analog/digital-input-port>,v=<value>,
event=underTheLevel

ここで、<送信先端末識別子>と<自端末識別子>は、8byte の端末識別子を 16 進で表したものである。

● set <id> sendif <alalog//digital-input-port>
setmode <mode>

上のコマンドで、内部変数 mode の値を設定するコマンドである。<mode>の値が内部変数 mode に代入される。

● set <id> i2c <address:0x**> <register:0x**>
<data-size> <value:0x**>

TWE-Lite の I2C 端子に接続された<address:0x**>で指定されたアドレスを持つ I2C スレーブデバイスの、<register:0x**>で指定されたレジスタに、大きさが<deta-size>バイトの<value:0x**>のデータを書き込むことを表す。

● get<id> i2c <address:0x**> <register:0x**>

TWE-Lite の I2C 端子に接続された<address:0x**>で指定されたアドレスを持つ I2C スレーブデバイスの、<register:0x**>で指定されたレジスタから値を入力し、その値を、このコマンドを送った端末に以下のコマンドで送り返す。

return a32=<送信先端末識別子>, from=<自端末識別子>,
port=<analog/digital-input-port>,v=<value>,
event=I2C

ここで、<送信先端末識別子>と<自端末識別子>は、8byte の端末識別子を 16 進で表したものである。

## 4. Bot/Gateway の開発

Bot/Gateway のハードウェアは、Raspberry Pi 3 と、モノワイヤレスの MoNoSTICK を組み合わせて開発している。図 3 に Bot/Gateway の例を示す。Raspberry Pi 3 は LAN 端子と WiFi アダプタを備えている。Bot/Gateway は、これらのどちらかで、インターネットを経由して、Wiki ネットワークと通信を行う。Raspberry Pi 3 に挿した MoNoSTICK を使って、無線センサネットワークに参加している TWE-Lite のセンサ端末と通信を行う。

Bot/Gateway の動作を、図 4 で示す。Bot/Gateway が起動すると、以下の動作を、繰り返し行う。

図 3. Raspberry Pi 3 を使って実現した、Bot/Gateway の例

1. 起動時に設定された Wiki Page に記載されているコマンド列やプログラムを読み込む。
2. このコマンド列やプログラムを実行する。
3. 2 の実行結果を元の Wiki page に書き戻す。

Bot/Gateway が実行するプログラムで、以下の関数が実行されることにより、無線センサネットワークを経由して、センサ端末にコマンドが送信され、センサ端末でコマンドそのコマンドが実行される。

ex("pi4j", "serial send ¥"<センサ端末が実行するコマンド>¥".")

先に述べたように、センサ端末でこのコマンドが実行されることにより、センサ端末は、コマンドによって設定された条件に沿って、以下の形式でセンサの値を送信する。

return a32=<送信先端末識別子>, from=<自端末識別子>,
port=<analog/digital-input-port>, v=<value>,
event=<この値を送信する要因>

Bot/Gateway がこのコマンドを受信したとき、<送信先端末識別子>が Bot/Gateway の MoNoSTICK の識別子と一致し、なおかつ、このコマンドが文字列 “return” を含んでいたら、return を取り除き、代わりに

device=sensorNetwork, Date=<受信日時>,

を先頭に付けて、この Bot/Gateway が定期手にコマンドやプログラムを読み込んでいる Wiki ページに書き込む。

Bot/Gateway と、MoNoStick 間の通信は USB serial インターフェースで行われ、k の通信を実現するために、Raspberry Pi のインターフェースを Java で扱う為に開発されている Pi4J を利用している。

## 5. 本 IoT システムの利用方法

### 5.1 Wiki サーバの設置と Wiki ソフトウェアの導入

Bot/Gateway が読み書きするための Wiki ページの Wiki サーバを、インターネットに直接接続できる場所で立ち上げる。Wiki サーバのソフトウェアとして、PukiWiki を使う。PukiWiki は日本で良く使われている Wiki ソフトウェアであり、多くの他の Wiki が Wiki ソフトウェアとは異なり、SQL サーバなどのデータベースソフトウェアを必要としない。既存の PukiWiki のサーバを利用することもできる。

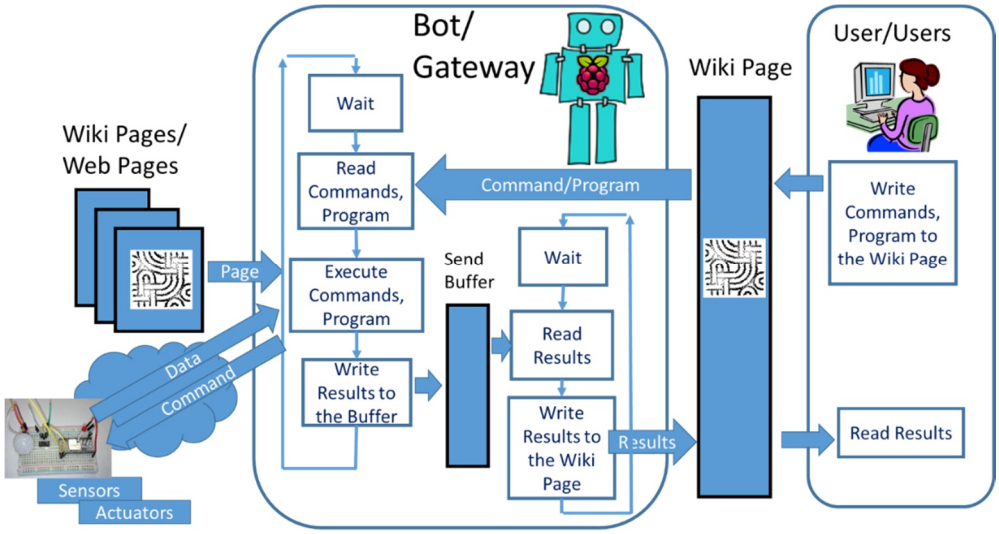


図 4. Bot/Gateway の動作

FrontPage

```
objectPage http://www.yama-lab.org/fwb4pi/index.php?SerialEx2 or http://www.yama-lab
device yamaRasPiDp9_1 or yamaRasPiDp9_2 start after no write for 10 min.
command: set readInterval=300000
command: set execInterval=0
command: program serial1
program: '
program: ex("pi4j", "serial send \"set a32=0x810008a8 sendif DI1 interval 31.\".")
program: ex("pi4j", "serial send \"set a32=0x8102dbdc sendif DI1 interval 39.\".")
command: end serial1
command: run serial1
result:
currentDevice="yamaRasPiDp1_1",Date=2017/1/27/ 0:9:11
```

図 5. センサ端末の設定を行う Wiki ページの例

```
program: ex("pi4j", "serial send \"set a32=0x810008a8 sendif DI1 interval 7000.\".")
program: ex("pi4j", "serial send \"set a32=0x8102dbdc sendif DI1 interval 39000.\".")
command: end serial1
command: run serial1
result:
device=sensorNetwork, Date=2017/1/30/ 0:41:10,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:41:19,  a32=0x810e0494, from=0x8102dbdc, port=DI1, v=1, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:41:41,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:41:58,  a32=0x810e0494, from=0x8102dbdc, port=DI1, v=1, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:42:12,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:42:37,  a32=0x810e0494, from=0x8102dbdc, port=DI1, v=1, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:42:43,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:43:3,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:43:10,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:43:16,  a32=0x810e0494, from=0x8102dbdc, port=DI1, v=1, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:43:21,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:43:26,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:43:31,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:43:38,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:43:45,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:43:52,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:43:56,  a32=0x810e0494, from=0x8102dbdc, port=DI1, v=1, event=interval.
device=sensorNetwork, Date=2017/1/30/ 0:44:0,  a32=0x810e0494, from=0x810008a8, port=DI1, v=0, event=interval.
```

図 6. 図 5 のプログラムが Bot/Gateway で繰り返し実行されていて、途中でプログラムを変更したとき

### 5.2 Bot/Gateway の設置

無線センサネットワークのセンサ端末と通信できる場所で、なおかつ、PukiWiki サーバとの間で通信できる場所に Bot/Gateway を設置する。Bot/Gateway が接続するネットワーク(TCP/IP ネットワーク)は、Bot/Gateway から PukiWiki サーバに接続可能であるなら、インターネットである必要はない。

### 5.3 無線センサネットワークの構築

使用するセンサ端末である TWE-Lite に、あらかじめ、3.2 で述べたコマンドを解釈実行するためのソフトウェアを導入しておく。このセンサ端末を、データ測定場所に設置する。センサ端末が直接 Bot/Gateway と通信できない場合は、この間に、最大３つの中継用 TWE-Lite を設置する。TWE-Lite への電源供給は、電池や商用電源や太陽電池などで行う。センサ端末の消費電力を抑えるためには、必要なとき以外は、センサ端末をスリープさせる手法が良くとられるが、本システムでは、Bot/Gateway から、いつ、コマンドがセンサ端末に送信されるかわからないので、センサ端末は常時受信可能な状態にしておく必要がある。

### 5.4 センサ端末の設定を行うコマンドとプログラムの書き込み

Wiki ページに、センサ端末の設定を行う為のコマンドとプログラムを書き込む。センサ端末の識別子がわからない場合は、id=* を記述することにより、すべてのセンサ端末にコマンドが送信される。なお、センサ端末から Bot/Gateway に同時に値が返送されると、Bot/Gateway がすべての値を受け取ることができない場合がある。図 5 に、センサ端末の設定を行う Wiki ページの例を示す。

図 5 において、

command: set readInterval=300000

の行は、このページを Bot/Gateway が読み込んだ後、その Bot/Gateway は、５分ごと(300000msec, 300sec)に、このページを読むことを表す。

command: set execInterval=0

の行は、ページの読み込みを行ったら、コマンドとプログラムを、読み込んだ後、１度だけ実行することを表す。execInterval=の右の数字が 0 より大きな整数である場合、その間隔(msec)で次のページ読み込みまで、繰り返し実行を行うことを表す。

command: program serial1

の行は、この行と、

command: end serial1

の間が、serial1 と名付けられたプログラムであることを表す。

program: ‘

の行の ‘ は、’より右がコメントであることを表す。

program: ex(“pi4j”,
“serial send ¥”set a32=0x810008a8
sendif DI1 interval 31000.¥”.”)

の行は、端末シリアル番号が 0x810008a8 であるセンサ端末において、入力ポート DI1 の値(0 または 1)を、Bot/Gateway に、31 秒ごとに送信するよう、指示することを表す。端末シリアル番号は、TWE-Lite の金属パッケージに記載されている。

program: ex(“pi4j”,
“serial send ¥”set a32=0x8102dbcd
sendif DI1 interval 39000.¥”.”)

の行は、端末シリアル番号が 0x8102dbcd であるセンサ端末において、入力ポート DI1 の値(0 または 1)を、Bot/Gateway に、39 秒ごとに送信するよう、指示することを表す。

command: run serial1

の行は、この場所で、プログラム serial1 の実行が開始されることを表す。

result:

の行は、この行以降から、各センサ端末が返送した、ポート DI1 の値などが記載されることを表す。

このページは、Bot/Gateway の実行が始まった後でも、利用者による書き換えが可能である。Bot/Gateway による書き込みと利用者による書き込みが重なる場合があるが、その時は、Wiki が自動的排他制御を行う。

### 5.5 Bot/Gateway の起動と実行ループの開始

Bot/Gateway のハードウェアの本体である Raspberry Pi において, Bot/Gateway のプログラムを起動し、この Bot/Gateway のコマンドやプログラムが書かれた Wiki ページの URL を与えて, Bot/Gateway の Wiki ページの読み込み・実行・書き込みのループを開始する。この実行ループが起動された後に、この Bot/Gateway が実行している Wiki のページのプログラムの、WSN のセンサ端末の設定部分が変更されると、それに従ってセンサ端末の設定が変更される。図 5 のプログラムの実行ループが開始された後、センサ端末 0x810008a8 におけるデータ入力、送信間隔を 31 秒から 7 秒に変更したときの、Wiki ページに書き戻されたデータの例を図 6 に示す。図 6 の横線の部分で設定が変更されている。

## 6. 関連研究

### 6.1 Galaxy Wiki

Galaxy Wiki[3] は Wiki に書かれたデータを解析するプログラムを Wiki 上に書いて、実行することができる Wiki である。我々の IoT システムでは、Wiki とそれを実行するハードウェア(Bot)が分離しているが、Galaxy Wiki の場合は、計算する部分もサーバが持っている。Galaxy Wiki は我々の IoT システムのように、実世界との相互作用を直接持つ

部分はない。

### 6.2 Xively

Xively[20]は、リアルタイムで物体検出装置の機械、建築、環境データからのリンクを共用し、ラベルを作成するネットワークサービスである。Xively は公開された API を持つ. このため, Xively にデータをアップロードしたり, アップロードされたデータを操作したりするプログラムを簡単に作ることができる.

本センサネットワークシステムも Xively と類似した機能を持つ. しかしながら, Xively の API は Xively のサイトでしか使えないのに対して, 本システムは, 特定のサイトだけでなく, 様々な PukiWiki のサイトで利用できる.

### 6.3 Twitter APIs

Twitter4J[19] は Java アプリケーションを Twitter と通信可能にする API である. これは Java プログラムがセンサデータを Twitter に書き込むことも可能にする. 我々の Bot は、Twitter4J を組み込んでおり, Twitter Wiki の読み書きだけでなく、Twitter の読み書きを行うことができる。

### 6.4 TinySCADA

Tiny Supervisory Control And Data Acquisition (TinySCADA) system[5] は Arduino と Google App Engine (GAE) を使った M2M システムである. このシステムと本システムとは, コンピュータによるシステム監視とプロセス制御を行うという点が似ているが。TinySCADA は GAE に依存している点で, 本システムと異なっている.

## 7. おわりに

我々が従来から開発していた, Wiki ページをデータベースとして使う IoT システムを拡張し, 無線センサネットワークのデータを入出力できるようにしたことについて述べた。IoT システムの利用場面において, センサ端末群から得られるデータを管理者が観察している最中に、特定のセンサ端末のみ、データを取得する時間間隔を途中で変えたくなる場合があるが、本システムは、このような要求を、物理的に、センサ端末がある場所に行かなくても、世界中、どこからでも、そのセンサ端末群を制御している Wiki ページの記述を変えるだけで、実現可能にする。今回、その具体例の一部を示すことができた。

WSN の TWE-Lite 間の通信はブロードキャストを使っており、信頼性がない。従って、WSN を使った IoT システムは信頼性が必要となるような分野で利用することができないが、複数の無線センサ端末を使って冗長性を持たせるなどにより、信頼性の欠如を補完できるのではないかと考えている。

本システムは NAT で守られた LAN 内で実行したり、Basic 認証が使えたりする以外は、なにもセキュリティ対策を行っていない.今後これらのやり残したことを実施していく予定である.

**謝辞** 本研究の一部は JSPS 科研費 16K00197, 15H03055 の助成を受けて実施しました。

## 参考文献

[1] John von Neumann, “First Draft of a Report on the EDVAC,” , Jun. 1945.
[2] Ward Cunningham. Wiki Wiki Web. http://wiki.c2.com/?WikiWikiWeb, 1995, As of Jan 22, 2017.
[3] WenPen Xiao, ChangYan Chi and Min Yang, “On-line collaborative software development via wiki”, Proceeding, WikiSym ’07, Proceedings of the 2007 International Symposium on Wikis, pp. 177-183, Montreal, Quebec, Canada, 2007.
[4] 山之上卓, 山本史弥, 小田謙太郎, 下園幸一, "Pukiwiki で Java プログラムの起動とそのデータ保存を可能とするシステムの試作", 情報処理学会 コンピュータと教育研究会第 109 回研究発表会 (CE 研究会), (2011 年 3 月)
[5] Miyanishi Yohtaro : A Prototype of SCAD A System using Cloud Computing Environment GAE , IEICE technical report 111(86), 7-12, 2011-06-10(2011)(in Japanese)
[6] Takashi Yamanoue, Kentaro Oda, Kochi Shimozono, "A Simple Application Program Interface for Saving Java Program Data on a Wiki," Advances in Software Engineering, Volume 2012 (2012), Article ID 981783, 9 pages,doi:10.1155/2012/981783, Hindawi Publishing Corporation, 2012.
[7] 山之上 卓, 白澤竜馬, 小田謙太郎, 下園幸一. Wiki と携帯型遠隔操作端末を使った情報セキュリティ監視システム, 情報処理学会研究会報告, Vol. 2012-IOT-16, No.35,(2012).
[8] Takashi Yamanoue, Kentaro Oda, Koichi Shimozono, A M2M system using Arduino, Android and Wiki Software, Proceedings of the 3rd IIAI International Conference on e-Services and Knowledge Management (IIAI ESKM 2012), pp.123-128, Fukuoka, Japan, 20-22 (Sep. 2012).
[9] Takashi Yamanoue, Kentaro Oda, Koichi Shimozono. A Malicious Bot Capturing System using a Beneficial Bot and Wiki, Journal of Information Processing(JIP), vol.21, No.2, pp.237-245(2013).
[10] Takashi Yamanoue, Kentaro Oda, Koichi Shimozono. An Inter-Wiki Page Data Processor for a M2M System, 4th International Conference on E-Service and Knowledge Management (ESKM 2013), Advanced Applied Informatics (IIAIAAI), 2013 IIAI International Conference on., pp.45-50, Matsue, Japan., (2013).
[11] 大澤文孝, TWE-Lite ではじめるカンタン電子工作, 工学社, 2014.
[12] 山之上卓, Twitter と Wiki を使った自動情報提示システム, 情報処理学会, 研究報告インターネットと運用技術（IOT）, vol. 2016-IOT-32, No.5, pp.1-7, (2016-03-03).
[13] 山之上卓, "Bot と Wiki を使った試験的な並列プログラミング", 情報処理学会, 研究報告インターネットと運用技術（IOT）, vol. 2016-IOT-34, No.2, pp.1-12, 2016-06-25.
[14] 平田篤, 藤田健吾, 伊勢本和広, 山之上卓, Wiki ページに書かれた R 言語のプログラムによるデータ解析を可能にした Bot の試作, インターネットと運用技術シンポジウム 2016 論文集, 情報処理学会シンポジウムシリーズ No. 2016, pp. 91-97, (2016-12)
[15] Android, https://www.android.com as of Jan. 30, 2017.
[16] Arduino, https://www.arduino.cc as of Jan. 30, 2017.
[17] PukiWiki, https://en.wikipedia.org/wiki/PukiWiki, as of Jan.. 30, 2017.
[18] Raspberry Pi, https://www.raspberrypi.org, as of Jan. 30, 2017.
[19] Twitter4J, http://twitter4j.org/ja/, as of Jan. 30, 2017
[20] Xively: https://xively.com/ as of Jan. 30, 2017.